

高级语言程序设计

第 2 章

数据与简单计算程序

华中师范大学物理学院 李安邦

上一章讲了快速入门编写**C/C++** 简单程序，
本章要学习一些基本的理论知识，然后再继续学习
如何编写简单计算程序。

第 2 章 数据与简单计算程序

2.1 计算机中的数值表示与存储

2.2 基本字符、名字表示、标识符和关键字

2.3 常用数据类型

2.4 运算符、表达式与计算

2.5 数据输出与简单数据输出程序

2.6 数学函数及其使用

2.7 Dev-C++中的辅助编辑功能

2.1 计算机中的数值表示与存储

用计算机处理信息，需要：

- 在计算机里存储和处理信息：就要求确定信息在计算机内部的表示方式，
- 将信息送给计算机，并能取得处理结果：能完成外部信息和计算机内部信息间的转换。
- 计算机处理信息的基础是把所有信息都用符号编码（数字化），为此需要：**1**、为数值确定一种计算机内部的表示方式；**2**、各种信息都确定相应的编码规则。

数据在计算机中的表示

- **计算机信息处理**是将信息转换为只有计算机本身才能识别和处理的计算机数据，即**机内数据**进行的。

数据的定义是广义的。包括字符、语言、图表、影像等。

- 数据有两种形态：
 - ◆ 人类可读形式的数据（人读数据）
 - ◆ 机器可读形式的数据（机读数据），用二进制的形式。

2.1.1 进位计数制

- 记数法：采用一组特定符号和一套统一规则表示数值的方法。最常用的是进位记数制。
- 进位记数制：就是将特定的数字符号（数码）顺序排列，根据各符号在序列中的位置确定其权重，并基于这种权重表达数值的方法。
- 用来表示数的数码的集合称为**基**，集合的大小称为**基数**。

十进制数：10个数码（0， 1， 2， 3， 4， 5， 6， 7， 8， 9），
基数为10，逢10进1

位权：某种进位计数制不同位置上数字的**单位值**，位置不同显示的数值大小不同。

在十进制中，10的整幂次方称为**10进制数的权**。

位权表示法：任何进位制数的值都可以表示成按照位权展开的多项式之和的形式。

例： $123.45 = 1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0 + 4 \times 10^{-1} + 5 \times 10^{-2}$

数位不同，权值不同。

按权展开通式：

$$(N)_{10} = \sum_{i=-m}^{n-1} a_i \times 10^i$$

其它进制

其它进制的计数规律可看成是十进制计数制的推广，
对任意进制 R ，数 N 可以表示成按权展开式：

$$(N)_R = (a_{n-1} a_{n-2} \dots a_1 a_0 \cdot a_{-1} a_{-2} \dots a_{-m})_R$$

$$\begin{aligned} (N)_R &= a_{n-1} \times R^{n-1} + a_{n-2} \times R^{n-2} + \dots + a_1 \times R^1 + a_0 \times R^0 \\ &\quad + a_{-1} \times R^{-1} + a_{-2} \times R^{-2} + \dots + a_{-m} \times R^{-m} \end{aligned}$$

$$= \sum_{i=-m}^{n-1} a_i \times R^i$$

1. R=2 二进制

数码个数 2 个: **0, 1**

计数规律: **逢二进 1, 借一当 2**

例: $(101.01)_2 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2}$

2. R=8 八进制

数码个数 8 个: **0, 1, 2, 3, 4, 5, 6, 7**

计数规律: **逢八进 1, 借一当 8**

例: $(176.5)_8 = 1 \times 8^2 + 7 \times 8^1 + 6 \times 8^0 + 5 \times 8^{-1}$

3. R=16 十六进制

数码个数 16 个: **0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F**

计数规律: **逢十六进 1, 借一当 16**

例: $(FA1.C)_{16} = F \times 16^2 + A \times 16^1 + 1 \times 16^0 + C \times 16^{-1}$



几种常用数制的表示方法：

十进制	二进制	八进制	十六进制
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

二进制系统

- 计算机数据采用二进制数表示。
- 原因：
 - (1) 设计可行：用任何一种能表示两个状态的物理量就可以表示 **0** 和 **1**。
 - (2) 算术运算和逻辑运算都相当简易
 - (3) 系统可靠

数制转换

二、八、十六进制转换为十进制：按权展开

例1: $(11011.11)_2 = (\textcolor{red}{27.75})_{10}$

$$= 1 \times \textcolor{red}{2^4} + 1 \times \textcolor{red}{2^3} + 0 \times \textcolor{red}{2^2} + 1 \times \textcolor{red}{2^1} + 1 \times \textcolor{red}{2^0} + 1 \times \textcolor{red}{2^{-1}} + 1 \times \textcolor{red}{2^{-2}}$$

$\textcolor{red}{16} \quad \textcolor{red}{8} \quad \textcolor{red}{0} \quad \textcolor{red}{2} \quad \textcolor{red}{1} \quad \textcolor{red}{0.5} \quad \textcolor{red}{0.25}$

例2: $(321.4)_8 = (\textcolor{red}{209.5})_{10}$

$$= 3 \times \textcolor{red}{8^2} + 2 \times \textcolor{red}{8^1} + 1 \times \textcolor{red}{8^0} + 4 \times \textcolor{red}{8^{-1}}$$

$\textcolor{red}{192} \quad \textcolor{red}{16} \quad \textcolor{red}{1} \quad \textcolor{red}{0.5}$

十进制转换为二、八、十六进制：整数相除，小数相乘

除基取余，
商零为止

乘基取整，满足
精度要求为止

例1: $(60.6875)_{10} = (\quad)_2$

2	60	... 余 0	低位
2	30	... 余 0	
2	15	... 余 1	
2	7	... 余 1	
2	3	... 余 1	
2	1	... 余 1	
2	0		高位

$(111100)_2$

0.6875
× 2
1.375
× 2
0.75
× 2
1.50
× 2
1.00

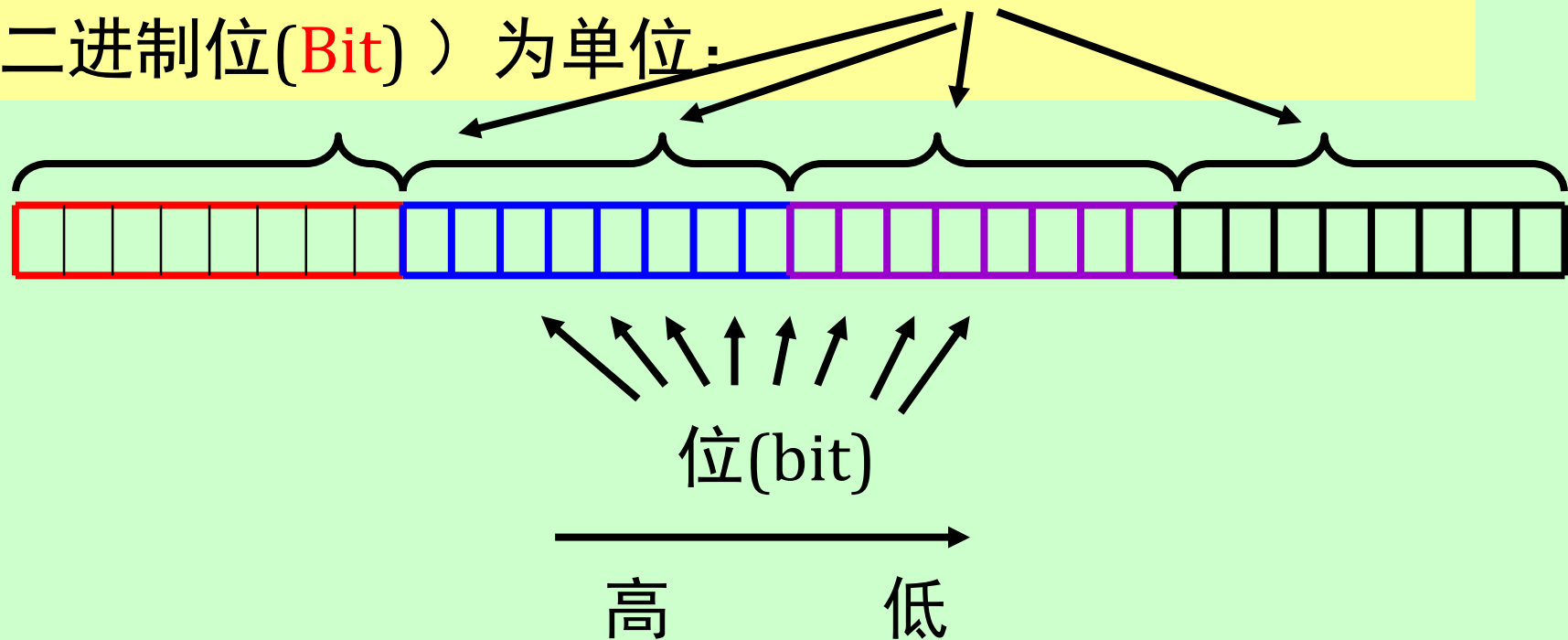
$(0.1011)_2$

数据单位

最小单位：**位**（**Bit**，比特），二进制的¹一个数位。

只能表示为 **0** 或 **1**

计算机的内部存储与操作常以**字节(Byte)**（**8** 个二进制位(**Bit**)）为单位：



其它存储单位

- 千字节(KB) $1\text{KB} = 2^{10}\text{ Bytes} = 1024\text{ Bytes}$
 $\approx 1000\text{ Bytes}$
- 兆字节(MB) $1\text{MB} = 2^{10}\text{ KB}$
- 吉字节(GB) $1\text{GB} = 2^{10}\text{ MB} = 2^{20}\text{ KB} = 2^{30}\text{ Byte}$
- 太字节(TB) $1\text{TB} = 2^{10}\text{ GB}$

小练习

1. 某文件有 100M，能否存储到 4G 的优盘中？
2. 2G 的 MP3 播放器大约能存多少 MP3 音乐文件（平均每个 5M）？
3. 2T 硬盘能存储多少 DVD 高清影片（平均每个 4G）？

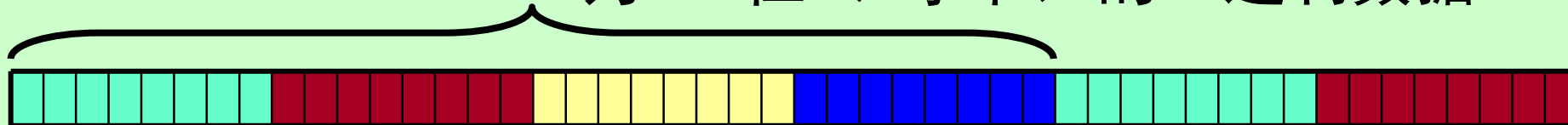
字(Word): 是计算机进行数据存储传送和处理的数据信息运算单位, 是由若干字节组成。

每个字所占的位数称之为**字长**。

字长越长, 处理速度就越快。



32位 CPU 在单位时间内能处理字长为 32 位 (4字节) 的二进制数据



16位 CPU



数据表示

计算机中的数据分为两大类：

（1）数值型数据：可进行算术运算的数据，
如：3.14，-1000。

两个问题：正负号的表示、小数的表示

（2）非数值型数据：不参与算术运算，
如：“ABC”、“国家”、“868”。

问题：如何用二进制数表示？

带符号数的代码表示

一、符号数

1.真值：在数值前加“+”号表示正数；
在数值前加“-”号表示负数。

2.机器数：把正负符号数值化的表示方法。

最高位用“0”表示正数，用“1”表示负数。

例：	真值(十进制)	真值(二进制)	机器数
	+9	+1001	01001
	-9	-1001	11001

符号位

原码表示法

原码表示法的特点

- (1)直观易辨认；
- (2)有2个0： $[0] = 00000 = 10000$
- (3)符号不参与运算；
- (4)减法运算不方便

反码 和 补码

其符号位规则相同，数值部分的表示形式有差异。
克服了原码表示法的缺点。但规则稍复杂。（略）

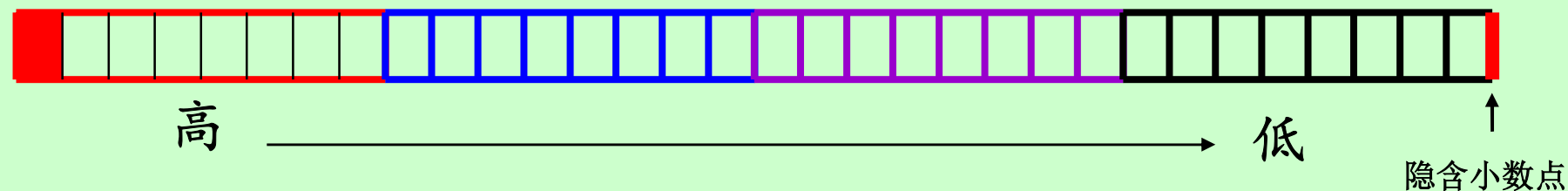
数值型数据

数据长度以**字节**为单位（长度为**8**位、**16**位或**32**位）

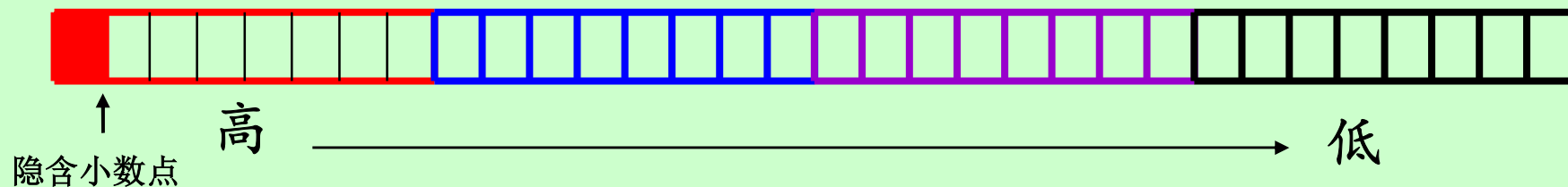
处理**小数点**的方法有两种：**1、定点数**；**2、浮点数**

定点数：

小数点隐含固定在**机器数的最低位**之后：



或 隐含固定在**机器数的符号位**之后：



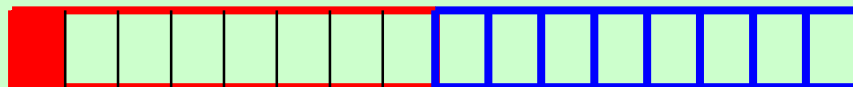
- 用定点数表示小数点简单直观，但数值取决于给定的字节数：



1字节： 有符号位： $-2^7 \sim +2^7 - 1$;
无符号位： $0 \sim +2^8 - 1$



2字节 有符号位： $-2^{15} \sim +2^{15} - 1$;
无符号位： $0 \sim +2^{16} - 1$



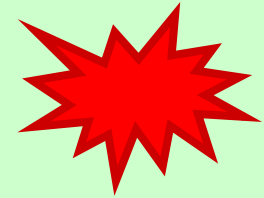
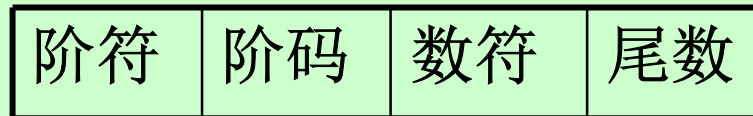
4字节 有符号位： $-2^{31} \sim +2^{31} - 1$;
无符号位： $0 \sim +2^{32} - 1$



总之，要注意表示的范围，避免溢出。

浮点数是由阶码和尾数两部分组成

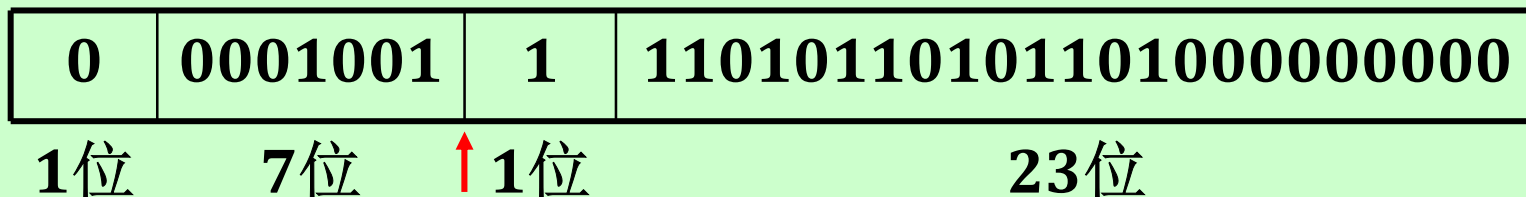
类似于科学计数法（例： $10^{+3} \times -1.2$ ）



例如：

$$-110101101.011010000000000 = 2^{+1001} \times -0.11010110101101$$

(规格化：尾数最高位为1)



浮点数由 32 位（4 字节）存储

- 需要注意的是，在计算机中用二进制浮点数表示小数，可能产生**浮点误差**，即在计算机中保存的小数与我们实际写出的十进制小数之间有误差。
- **（1）有限位的十进制小数，用二进制表示时可能会是一个无限位的小数。**例如十进制数**0.9**转化成二进制表示时将得到一个无限循环小数**0.1110011001100110011...**
- **（2）计算机浮点数的精度有限**，例如，按上面所说的形式，用**4**个字节存储的浮点数只能表示十进制的最多**7**位有效数字，用**8**个字节存储的浮点数可以表示十进制**15~16**位有效数字。**超过有效数字之后的数位只能忽略。**例如，上面十进制数**0.9**在计算机中用二进制表示，受保存精度所限，有效数字之后的部分就要忽略。这样，用**4**个字节存储的浮点数表示时对应于**0.89999998**，用**8**个字节存储的浮点数表示时对应于**0.900000000000000002**。

字符数据的信息编码

- 字符是计算机中使用最多的数据形式之一，在计算机中要为每个字符指定一个确定的编码，作为识别与使用这些字符的依据。
- 字符数据在计算机内也是用二进制数表示。
- 西文字符数量少，只需要一种编码就够了。普遍采用的是 **ASCII 码** (美国信息交换标准代码)。
- 中日韩文字（以及世界其它各国的字符）用双字节或四字节编码。

西文字符普遍采用的是 **ASCII 码** (美国信息交换标准代码)。

ASCII 值	字符	ASCII 值	字符	ASCII 值	字符	ASCII 值	字符
0	NUL	32	(space)	64	@	96	`
1	SOH	33	!	65	A	97	a
2	STX	34	"	66	B	98	b
3	ETX	35	#	67	C	99	c
4	EOT	36	\$	68	D	100	d
5	ENQ	37	%	69	E	101	e
6	ACK	38	&	70	F	102	f
7	BEL	39	,	71	G	103	g
8	BS	40	(72	H	104	h
9	HT	41)	73	I	105	i
10	LF	42	*	74	J	106	j
11	VT	43	+	75	K	107	k
12	FF	44	,	76	L	108	l
13	CR	45	-	77	M	109	m
14	SO	46	.	78	N	110	n
15	SI	47	/	79	O	111	o
16	DLE	48	0	80	P	112	p
17	DC1	49	1	81	Q	113	q
18	DC2	50	2	82	R	114	r
19	DC3	51	3	83	X	115	s
20	DC4	52	4	84	T	116	t
21	NAK	53	5	85	U	117	u
22	SYN	54	6	86	V	118	v
23	TB	55	7	87	W	119	w
24	CAN	56	8	88	X	120	x
25	EM	57	9	89	Y	121	y
26	SUB	58	:	90	Z	122	z
27	ESC	59	;	91	[123	{
28	FS	60	<	92	/	124	
29	GS	61	=	93]	125	}
30	RS	62	>	94	^	126	~
31	US	63	?	95	_	127	DEL

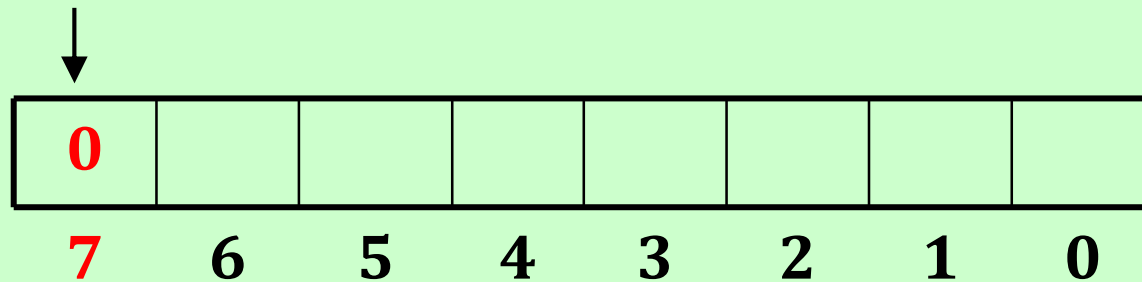
控制符

编码为 33 - 126
的字符为可打印
的人读字符

控制符

标准 ASCII 码共 128 个字符，只需 7 位二进制数编码（ $2^7=128$ ）

计算机的内部存储与操作常以字节（8 个二进制位）为单位，因此一个字符实际上用一个字节表示，最高位 d_7 置为 0。

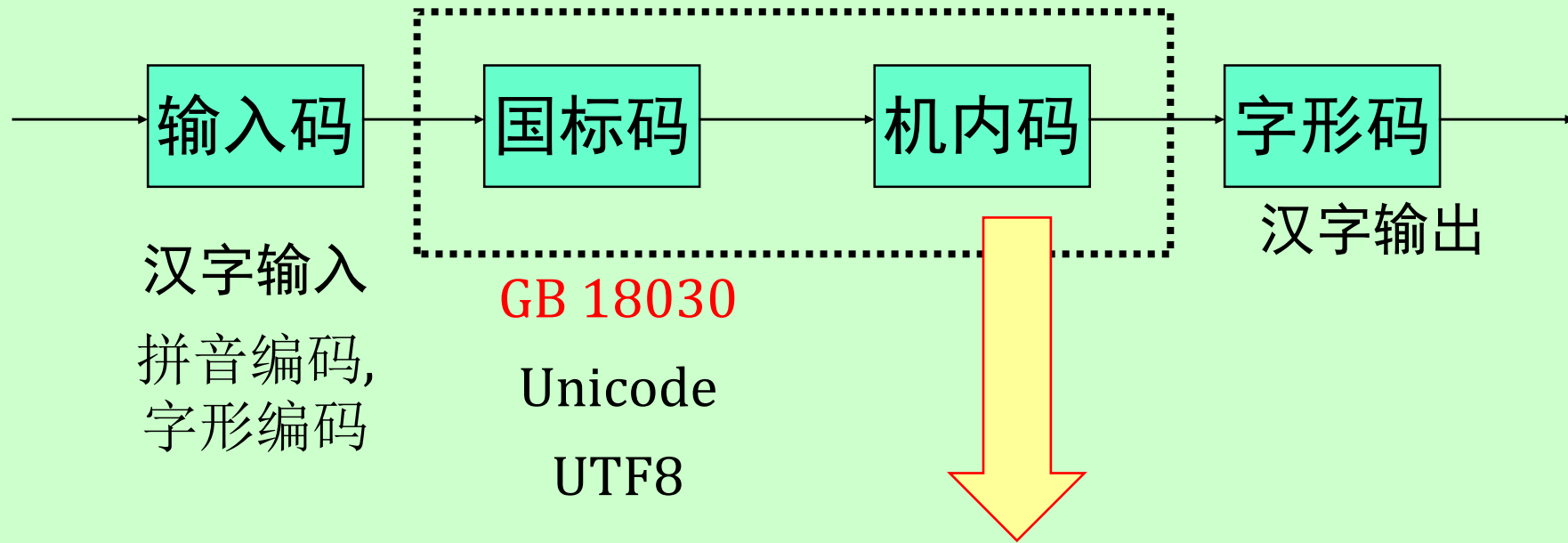


$$32 = (0010)_2$$

ASCII编码表

$b_7b_6b_5b_4$ $b_3b_2b_1b_0$	0000	0001	0010	0011	0100	0101	0110	0111
0000	NUL	DLE	SP	0	@	P	`	P
0001	SOH	DC1	!	1	A	Q	a	Q
0010	STX	DC2	"	2	B	R	b	R
0011	ETX	DC3	#	3	C	S	c	S
0100	EOT	DC4	\$	4	D	T	d	T
0101	ENG	NAK	%	5	E	U	e	U
0110	ACK	SYN	&	6	F	V	f	V
0111	BEL	ETB	'	7	G	W	g	W
1000	BS	CAN	(8	H	X	h	X
1001	HT	EM)	9	I	Y	i	Y
1010	LF	SUB	*	:	J	Z	j	Z
1011	VT	ESC	+	;	K	[k	{
1100	FF	FS	,	<	L	\	l	
1101	CR	GS	-	=	M]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

汉字编码



采用与 ASCII 码相容的 16位码，

每个汉字占两个字节，最高位设为“1”。



第 2 章 数据与简单计算程序

2.1 计算机中的数值表示与存储

2.2 基本字符、名字表示、标识符和关键字

2.3 常用数据类型

2.4 运算符、表达式与计算

2.5 数据输出与简单数据输出程序

2.6 数学函数及其使用

2.7 Dev-C++中的辅助编辑功能

2.2 基本字符、名字表示、标识符和关键字

- C/C++ 程序是用基本字符写出的符合规定形式的字符序列。
- 基本字符包括：
 - ◆ 数字（0～9）
 - ◆ 大小写字母（a～z，A～Z）和下划线
 - ◆ 标点符号
 - ◆ 特殊字符：空格、换行、制表符（空白字符），起分隔作用。增删空白一般不影响程序的意义
- 应该利用空白字符排列程序格式，使程序的形式更好地反映程序结构和它所实现的计算过程。

标识符



- 程序有许多对象（例如数据、变量和函数），需要命名进行区分，给对象所命的名称就叫做**标识符 (identifier)**。——通过每个名称能唯一确定一个对象。
- 通过给各种对象命名，建立起**定义**和**使用**的联系。
- 形式：**字母、下划线或数字的连续序列，必须以字母或下划线“_”开始，中间不能含有空格。**
- **标识符对字母大小写敏感：a 和 A 是不同字母；ABC、Abc、AbC 和 abc 是4个不同标识符。**

- 合法标识符示例：

abcd sin Beijing C_Programming a3b06

while sx211_12a abc_ _f2048 ____

- 不合法标识符：

2A A-B area of circle M.D

- 合法的标识符可以用于表达式或语句：

例：**x3 + 5 ab_400 + xy_ / x**

- 尽量采用能说明程序对象意义的标识符。


例如：**min** 表示最小值，**average** 表示平均值，**day** 表示日期。

- 关键字（Keywords）：特殊标识符集（ANSI C 共32个，C99有扩充），有特定意义，不允许用户再用作其它用途。

**auto break case char const continue
default do double else enum extern
float for goto if int long register
return short signed sizeof static struct
switch typedef union unsigned void
volatile while**

- 以下划线开始的标识符保留给系统使用。用户不要自定义这种标识符。

2.3 数据、类型和简单程序

- 类型（**type**）是计算机科学的一个核心概念。
- 一个类型是程序里可用的一个数据对象集合。 
- C/C++ 基本数据类型包括**字符**、**整数**、**实数**类型等
- 同类型所有数据对象的性质相同，采用统一书写形式，同样编码方式，能做同样操作。
- 数据都属于特定类型。基本类型用定长二进制编码表示。确定了该类型的可能范围。

- 类型名：基本类型的类型名由一个或者几个关键字组成，例如：

int long int

float double long double

- 本章介绍几个最常用的类型，以尽早进入程序设计的主题。有关 C 语言基本类型的完整介绍见后。
- 文字量：源程序里直接写出的数据。（常常需要经过转换才变成实际含义）
- 整数类型的文字量（简称整数）：写在程序里的整数类型的数据。其他情况类似。

2.3.1 几个常用类型

整数类型和整数

- C/C++ 语言里有多个整数类型
- 不同整数类型所用编码位数可能不同。常用：
 - ◆ 整数类型: **int**
 - ◆ 长整型类型: **long int**, 简写**long**
- **int** 和 **long** 都是关键字
- **long** 表示范围等于或大于 **int** 的表示范围。具体由 C /C++ 系统确定



整型文字量（整数）写法

常用十进制写法，首字符非 **0**（除非 本身是 **0**）

例：**1234 0 768 2047 1999 2000**

长整数写法：加后缀 **l** 或 **L**（不能有间隔）：

123L 304l 25278L 1l 0L ⇐ 体会“文字量”

小写 **l** 易与数字 **1** 混淆，建议用大写

整数可以加正负号

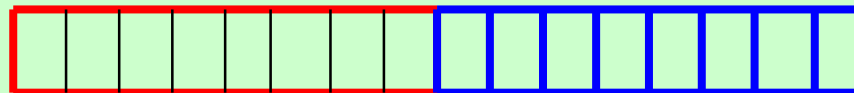
整数表示范围

- 编码长度（表示范围）由具体 **C** 系统确定。
系统规定：**long** 的表示范围不小于 **int**。
- 例：现在的个人计算机上的**整数**用 **32** 位存储：



若首位表示正负号，则表示范围为： $-2^{31} \sim 2^{31}-1$ ，
即： **$-2147483648 \sim 2147483647$**

一些早期的 **C** 系统中**整数**用 **16** 位存储：



表示范围是： **$-32768 \sim 32767$**

* 整数的八进制和十六进制表示

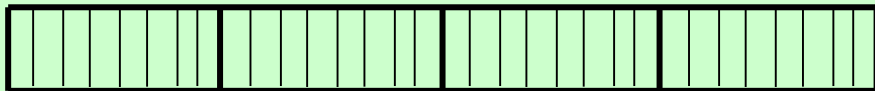
- 整数可以用八进制和十六进制形式写
- 八进制：**0**开头的数字序列，数字仅用**0~7**
0123 06254 0531 0765432L
- 十六进制：**0x** 或 **0X** 开头的数字序列。用字母**a~f** 或 **A~F** 表示其余6个数字
0x2073 0xA3B5 0XABCD 0xF0F00000L
- 文字的十进制、八进制和十六进制表示形式只是整数的不同书写形式，是为编程方便。

实数类型和实数的表示

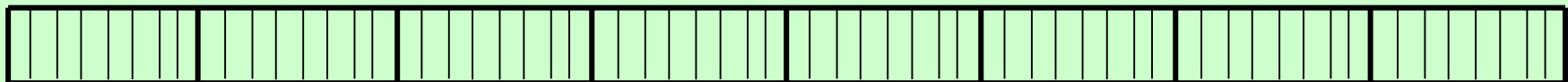
- 单精度浮点数类型（浮点类型） **float**
- 双精度浮点数类型（双精度类型） **double**
- 长双精度类型 **long double**
- 字面量：浮点数/双精度数/长双精度数

常用的标准表示：

- 浮点数用 **32** 位表示，约 **7** 位十进制有效数字



- 双精度数用 **64** 位表示，约 **16** 位十进制有效数字



- 长双精度数用 **64** 位或 **80** 位表示（由系统定义）

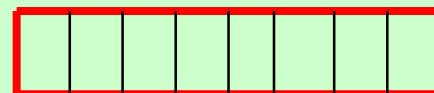
实数写法

- 基本实数类型是 **double**: 数字序列，需包含小数点 “.”（可以是首/末字符）或指数部分
- 指数: **e/E** 开头数字序列（可带符号），以**10**为底。可同时有小数点和指数。
- 例: **3.2** **.1** **3.** **2E-3** **2.45e17**
 2×10^{-3} 2.45×10^{17}
- **float** 加后缀 **f** 或 **F**, **long double**加 **L**
- 实数前可以有正负号
- 整数类型和实数类型统称**算术类型**。

字符类型和字符的表示

- 字符类型用于输入输出（I/O）或文字处理
- 常用类型 **char**，包括所用机器的字符集的所有字符。字符在内部用对应编码表示。

- 一个 **char** 占一个字节：



- 微机常用 **ASCII** 字符集，包含**128**个字符。有时用扩展**ASCII**字符集，**256**个字符。

- 字符的文字量：单引号括起的一个字符：

'1' 'a' 'D'

- 常用特殊字符（换意序列写法）：

换行字符 '**\n**' 双引号字符 '**\\"'** 单引号字符 '**\''**

反斜线字符 '****' 制表符 '**\t**'

字符串

- 表示一串字符的数据描述形式。常用于输入输出。
- 文字量：用一对英文**双引号**把一串字符括起来。
- 特殊字符用换意序列表示。

例： **"CHINA" "Welcome\n"**
 "He said: \"Ok.\"\n"

第1章的简单 C/C++ 程序里有：

```
cout << "Hello, world!" << endl;
```

字符串里的空格是实际内容（“有意义”）

字符串主要用于输入输出：

```
cout << "Hello, world!" << endl;
```

等价写法：

```
cout << "Hello, world!" << "\n";
```

```
cout << "Hello, world!" << '\n';
```

```
cout << "Hello, world!\n";
```

在字符串文字量的中间不能换行，否则编译会出错。
如果需要写很长的字符串，可以将其分开写成几个字符串。

```
cout << "C++ (\nsee plus plus\n") is a very popular"  
      "programming language developed by Bjarne "  
      "Stroustrup in 1979."
```

第 2 章 数据与简单计算程序

2.1 计算机中的数值表示与存储

2.2 基本字符、名字表示、标识符和关键字

2.3 常用数据类型

2.4 运算符、表达式与计算

2.5 数据输出与简单数据输出程序

2.6 数学函数及其使用

2.7 Dev-C++中的辅助编辑功能

2.4 运算符、表达式与计算

- 表达式：描述计算的最基本结构。由被计算对象（如文字量）和运算符“按一定规则”构成。
- 描述运算的特殊符号，所有运算符都用一个或两个特殊字符表示（有一个例外）
- 本节将介绍各种算术运算符的形式和意义，介绍如何用它们构造算术表达式。
- 还要介绍一些与运算符、表达式和表达式所描述的计算有关的重要问题。
- 理解这些问题，才能正确写出所需的表达式。

2.4.1 算术运算符

- | 运算符 | 使用形式 | 意义 |
|-----|----------|-----------|
| + | 一元和二元运算符 | 一元正号，二元加法 |
| - | 一元和二元运算符 | 一元负号，二元减法 |
| * | 二元运算符 | 乘法运算 |
| / | 二元运算符 | 除法运算 |
| % | 二元运算符 | 取模运算（求余数） |
- 由 +/- 的上下文可确定是“一元”还是“二元”。
 - % 只能用于整型（常用于判断是否能整除），其余可用于各种算术类型。

2.4.2 算术表达式

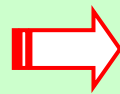
- 算术表达式 形式与数学的算术表达式类似:

$$-(28 + 32) + (16 * 7 - 4)$$

$$25 * (3 - 6) + 234$$

- 平方和立方通常写成连乘: $10*10$, $2.1*2.1*2.1$
- 算术表达式中只能使用圆括号作为多级括号。
- 源程序是文本文件, 只能从左到右顺序地书写, 不像数学中的公式那样具有二维的形式。所以要学会把数学表达式改写为源程序中的表达式:

$$\frac{8.5}{4} + \frac{3 + 2 \times 6}{5}$$



$$8.5 / 4 + (3 + 2 * 6) / 5$$



在程序中的适当位置写好所需的算术表达式。计算机运行程序时遇到一个表达式，就会完成该表达式描述的计算，按照程序中的要求去使用得到的值。



【例2-1】把上面举例的三个算术表达式写在程序中，把它们的值用“**cout <<**”输出。

```
#include <iostream>
using namespace std;
int main () {
    cout << -(28 + 32) + (16 * 7 - 4) << endl;
    // 输出表达式的值
    cout << 25 * (3 - 6) + 234 << endl;
    cout << 8.5 / 4 + ( 3 + 2 * 6) / 5 << endl;
    return 0;
}
```

多多练习！

表达式是由被计算对象和运算符构成的。但是不要死抠概念。例如：

- **3+2**

- **3**

- 前面说的数据存储方式、数据类型与表达式有什么关系？

- 当你按照前面的“数据类型”中所说的的方式书写表达式时，计算机内部就会把它们按照相应的类型来处理。

- **3+2L 5+6.2**

2.4.3 表达式的求值

- 表达式计算又称“表达式求值”。
- 一个表达式的意义就是它求出的值。
- 语言明确规定了表达式计算过程。包括几方面：
 - 优先级
 - 结合方式
 - 括号
 - （多个）运算对象的求值顺序

- 优先级：运算符在表达式中相邻出现时，优先级高的运算符先算。（附录A 运算符表）

- 算术运算符分三个优先级：

一元 $+$, $-$ (高) $*$, $/$, $\%$ (中) 二元 $+$, $-$ (低)
(正负号最高，乘除和取余次之，加减最低)

例： $5 / -3 + 4 * 6$ (符合数学学习惯)

- 结合方式：同优先级运算符相邻时的计算顺序

- 一元算术运算符自右向左结合，二元算术运算符自左向右结合。 (符合数学学习惯)

例： $166 / 8 * 5 / 3$

- 括号：改变计算顺序，括号括起的部分先算。

$$-(((2 + 6) * 4) / (3 + 5))$$

- 括号是控制计算顺序的手段。【符合数学习惯】

-
- 有关格式的建议：

- 1、表达式较复杂时，应适当加括号，以利阅读
- 2、表达式可换行，应采取某种对齐方式，以利于理解，出错也容易发现和改正

正例：

$$(2 + 3.23/2.15 - 0.6) \\ * (1.66 + 2.87/4.13 - 2.83)$$

反例：

$$(2 + 3.23/2.15 - 0.6) * (1.66 \\ + 2.87/4.13 - 2.83)$$

- 对于复杂的表达式，分析计算过程时需要考虑上述的**优先级**、**结合方式**和**括号**这三方面。
- 编程时如何正确地写出复杂表达式？
- 牢记：
乘除取余优先于加减，其余的加括号

运算对象的求值顺序问题

问：表达式 $(5 + 8) + (6 + 4)$ 中，加号左右两边的 $(5 + 8)$ 和 $(6 + 4)$ 中哪个先做？

答：C/C++ 语言对此问题无规定。

语言里可以写出对求值顺序敏感的表达式（数学中无此问题）。程序中不要写依赖特殊计算顺序的表达式，那样将无法保证得到的结果。

至此，我们已经可以
写出许多程序了！

```
#include <iostream>  
using namespace std;
```

```
int main () {  
    cout << "Hello, world!" << endl; //屏幕输出字符串  
    cout << 8.5 / 4 + ( 3 + 2 * 6 ) / 5 << endl; //表达式求值  
    return 0;  
}
```


2.4.4 计算和类型

- 表达式计算中有许多与类型有关的问题。

(一) 同类型数据计算的顺序

- 对属于同一类型的一个或两个数据使用算术运算符，计算结果仍是该类型的值。
- 影响：整数的除法是整除，得到整数的商，余数自动丢掉。
- 对比（用 **cout <<** 测试）：

$2/5 \leftrightarrow 3/5$

$10/5 \leftrightarrow 10/4$

$1 / 5 * 5 \leftrightarrow 1 * 5 / 5$

例

算术计算的溢出



- (1) 每个类型有明确取值范围
- (2) 计算有确定的结果类型
- 计算中结果超出类型表示范围称为溢出。溢出后的计算没有意义。
- **C 程序对溢出不报错。**

例：在很旧的计算机系统上，**int** 由**16**位表示(可表示的数值范围为 **-32768 ~ +32767**)，则下面表达式会溢出：

32766 + 18

如果可能，应选择合适类型。如改为：

32766L + 18L

- 当代的计算机系统中，**int** 用 **32** 位表示，可表示的数值范围为：**-2147483648 ~ 2147483647**
- 实数计算也可能发生溢出（上溢和下溢）。

(三) 混合类型计算和类型转换

- 运算对象类型不同时形成混合类型计算：

3.27 + 201

- 程序自动将运算对象转换到相同类型的值，然后计算。是自动类型转换。不需要明确写出。
- 原则：把表示范围小的类型的值转换为表示范围大的类型的值。

从小到大：**int long float double long double**

- 由原类型的值产生出新值后参与计算。可以借用来避免整数相除丢失小数部分和避免整数溢出。

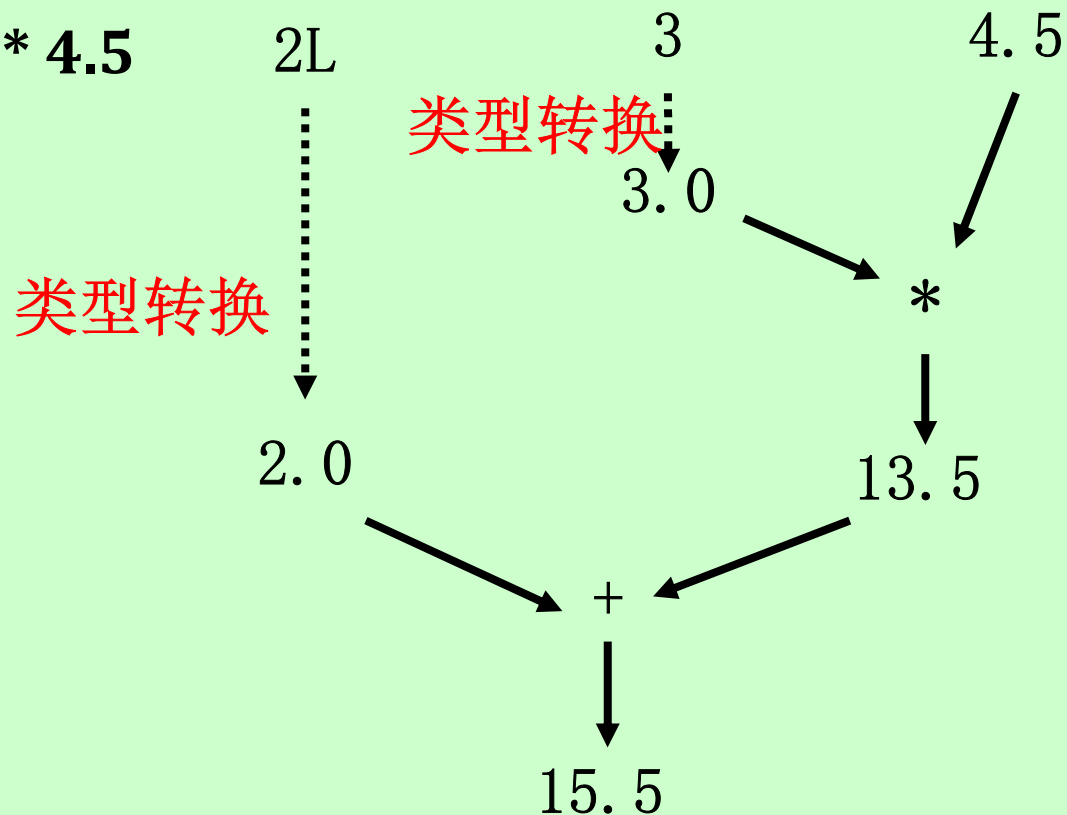
例：**2.0/5 2/5.0 2./5 2147483647 + 1.0**

写程序或读程序都应注意:



- 表达式中计算对象的类型
- 子表达式计算结果的类型
- 哪些地方会发生类型转换, 怎么转换

例: 表达式 $2L + 3 * 4.5$
的计算过程:



在计算数学表达式时，需要根据所需精度考虑合理的写法。例如：

$$1 + \frac{2}{3 + 4/5}$$

保留小数：

$$1 + 2 / (3 + 4.0 / 5)$$

$$1 + 2 / (3 + 4 / 5.)$$

$$1.0 + 2.0 / (3.0 + 4.0 / 5.0)$$

丢弃小数：

$$1.0 + 2 / (3 + 4 / 5)$$

$$1 + 2.0 / (3 + 4 / 5)$$

$$1.0 + 2.0 / (3.0 + 4 / 5)$$

三、显式类型转换

- 如果自动转换不符合需要，可要求做特定类型转换，称为强制转换或类型强制
- 写法：表达式前写括起的类型名。例：

(int)(3.6 * 15.8) + 4 //实数类型转为整型时丢掉小数部分

(double)2 / 3 //整型转换为 **double** 型，再进行计算

与类型转换有关的问题：

- 类型转换可能丢失信息。
- 若被转换值在结果类型里无法表示，结果无法预计。写强制转换时必须注意。
- 显式类型转换看作一元运算符，与其他一元运算符有同样优先级和结合方式。
- 类型转换是值转换，从一个数据值出发，产生另一类型的新值，原值不变。
- 数值类型间都可以转换。其他转换后面讨论。

第 2 章 数据与简单计算程序

2.1 计算机中的数值表示与存储

2.2 基本字符、名字表示、标识符和关键字

2.3 常用数据类型

2.4 运算符、表达式与计算

2.5 数据输出与简单数据输出程序

2.6 数学函数及其使用

2.7 Dev-C++中的辅助编辑功能

2.5 数据输出与简单数据输出程序

- **C/C++** 语言规范定义了一个很大的标准库，其中提供了许多常用功能，供我们在编程中使用。
- 每个 **C/C++** 系统都包含了标准库，程序中使用的基本输出功能也由标准库提供。
- **C++** 的输出功能基于流的概念，**cout** 就是一个标准输出流，程序对**cout** 的输出默认将送到屏幕或屏幕中特定窗口中显示。
- 经常用到输出 **endl**，它是一个由**C++** 内部定义的符号常量“换行符”，用于让光标移动于下一行进行输出。可以根据需要而输出**0**个、**1**个或多个。

- 标准输出流 **cout** 在标准库文件 **iostream** 中定义。要使用 **cout** 和相关功能，程序中必须写：

#include <iostream>

- 打开命名空间 **std**，使程序中能使用 **cout**：

using namespace std;

- 针对 **cout** 输出用 “<<”（称为**插入运算符**）描述，后跟一个表达式，表示要求将这个表达式的求值结果送入输出流 **cout**，输出到标准输出设备。
- 可以输出多个表达式的值，每个表达式之前都写 “<<”。

cout << "Hello, " << "world!" << endl;

- 用 **cout <<** 输出表达式的类型为**字符**、**字符串**或**整数**时，执行中将以自然的形式输出结果。
- 在输出各种**浮点数**类型的值时，**cout** 产生输出时能根据情况做些调整。会按默认精度，以小数形式或科学计数法输出。
- 有一组**I/O流的操纵符**用于说明特殊输出要求。要使用则需要源程序中写有：**#include <iomanip>**
- **setw(n)** 用于设定实际输出宽度为**n**个字符位置，实际输出在这段位置中居右对齐。
- **fixed** 将浮点数按照普通定点格式输出（不省略末尾的**0**）
- **scientific** 将浮点数按照科学计数法格式输出（带指数域）
- **setprecision(n)** 设置输出浮点数的精度为 **n**。

【例2-2】使用I/O流的操纵符进行输出 π 的近似值。

```
#include <iostream>
#include <iomanip>
using namespace std;
```

```
int main() {
    cout << "默认格式输出: \n" << 3.14159265 << endl;
    cout << "指定宽度输出: \n" << setw(12) << 3.14159
    cout << "指定精度和宽度输出: \n"
        << setprecision(8) << 3.14159265 << endl
        << setw(12) << 3.14159265 << endl << endl;

    cout << "默认格式输出: \n" << 314.15900 << endl;
    cout << "指定宽度输出: \n" << fixed << 1314.15900 << endl;
    cout << "科学计数法格式输出: \n" << scientific << 314.15900 << endl;
    cout << "默认格式输出: \n" << 314.15900 << endl;
    return 0;
}
```

默认格式输出:

3.14159

指定宽度输出:

3.14159

指定精度和宽度输出:

3.1415927

3.1415927

默认格式输出:

314.159

指定宽度输出:

314.15900000

科学计数法格式输出:

3.14159000e+002

默认格式输出:

3.14159000e+002

2.5.2 C语言中的 printf 函数

- 要想让C程序产生输出，需要使用C语言的标准函数库（标准库）。每个C语言系统都提供了标准库，包含许多常用函数。
- 常用输出函数：**printf**。它能够将信息送到标准输出（一般是屏幕或特定窗口）。基本使用形式：

printf (字符串);

- 函数调用形式：**函数名，括号，函数参数，多个参数用逗号分隔**
- 这称为**语句**：程序基本单位，以分号为结束符

printf 函数的详细使用功能:

printf(格式描述串, 其他参数)

第一个参数应是字符串，可以有其他参数。

- 如果“格式描述串”里没有 **%**，也没有其他参数，**printf** 输出格式描述串。

printf(格式描述串, 其他参数)

- 格式描述串中，以 % 开始的段意义特殊，称为“转换描述”：描述数据的输出转换方式
- 格式串的转换描述应与其他参数的个数一致。
- 每个转换描述说明一个参数的输出形式（转换方式）。
- 格式串里的普通字符直接输出，而转换描述用对应参数的转换结果替代。

输出整数： `printf(" %d + %d = %d\n", 2, 3, 5);`

输出长整数： `printf(" %ld + %d", 3L, 5);`

要熟记!

常用转换:

<u>转换描述</u>	<u>实现的转换</u>	<u>对应参数的类型</u>
%d	整数转换输出	int 类型
%ld	长整数转换输出	long 类型
%f	实数转换输出	double 类型
%Lf	实数转换输出	long double 类型
%s	输出字符串	字符串
%c	输出字符	字符 (编码)

例:

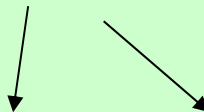
```
printf("len: %f, width: %f\n", 2.2, 3.5);
```

```
printf("Name: %s, gender: %c\n", "Tom", 'M');
```


- 在 % 和转换字符之间可写一个整数表示输出宽度。
- 对浮点数还可指定小数位数（例如，%8.4f 表示 8 位精度，其中 4 位小数）：

```
printf("%10s: %4d %8.4f\n", "Li Ming", 2, 100.2);
```

-
- 小技巧：要实现多行输出对齐，常常指定输出宽度，并用制表符分隔：



```
printf("%10s\t%4d\t%8.4f\n", "Li Ming", 2, 100.2);  
printf("%10s\t%4d\t%8.4f\n", "Tom", 10, 3.2);
```

2.5.3 简单计算程序

【例2-4】 计算半径为 6.5 厘米的圆球体积。

```
#include <iostream>
using namespace std;
int main() {
    cout << "Calculate sphere's volume." << endl;
    cout << "R = " << 6.5 << " cm\n";
    cout << "V = " << (3.1415927 * 6.5 * 6.5 * 6.5) * 4 / 3
        << " cm^3\n";
    return 0;
}
```

$$V = \frac{4}{3} \pi R^3$$

运行时输出：

V = 1150.349200 cm^3

这是最简单计算程序的框架，可替换其中的表达式。

- 或许有读者会想，既然题目只要求计算体积，只在**main**函数中简单写一条语句就可以了：
- **cout << (3.1415927 * 6.5 * 6.5 * 6.5) * 4 / 3;**
- 如果这样，程序运行时将只输出体积计算的结果：
- **1150.35**
- 显然，这样写可以减少语句条数和输出的内容，缩短程序的长度，确实也满足了题目的要求。但是我们不赞同这种极简的写法，而提倡让程序多输出一些信息。我们建议，在编程时，应当让程序合理地输出一些信息，帮助用户更好地理解程序运行的情况和计算的结果。

- 上面程序中还有一些细节值得注意：
- (1) 在字符串 "R = " 和 "V = " 尾部加了空格，"cm\n" 和 "cm^3" 头部加了空格，这就使输出数据的前后有空格，更美观易读。
- (2) 圆球体积的计算公式不能写成 “ $4 / 3 * (3.1415927 * 6.5 * 6.5 * 6.5)$ ”，那样会由于对 “ $4 / 3$ ” 求值得到1，产生较大的计算误差。
- (3) “cm^3” 是人们以纯文本方式书写带上标的度量单位cm³时的常见简写方式。这里的'^'只是字符串里的字符，并没有运算的意义；
- (4) 如果需要输出换行，可以任意选择使用endl或'\n'。

【例2-5】前面说过，“属于同一类型的数据进行算术运算时，计算结果仍然是该类型的值”，所以，表达式中出现整数相除的结果仍为整数。请编一个程序，验证下面情况或求出要求的结果：（1）表达式3/5和4/5的值都是0；（2）求出数学式 $3/5$ 和 $4/5$ 的浮点数近似值；（3）通过合理的表达式求出数学公式 $1 + \frac{2}{3 + 4/5}$ 的尽可能精确的值。

```
#include <iostream>
using namespace std;
int main() {
    cout << "整数相除: " << 3/5 << "\t" << 4/5 << endl << endl;
    cout << "混和类型计算: " << 3./5 << "\t" << 4/5.0 << endl << endl;

    cout << "正确书写表达式" << endl;
    cout << 1 + 2 / ( 3 + 4.0 / 5 ) << endl;
    cout << 1 + 2. / ( 3 + 4 / 5 ) << " wrong\n" << endl;
}
```

第 2 章 数据与简单计算程序

2.1 计算机中的数值表示与存储

2.2 基本字符、名字表示、标识符和关键字

2.3 常用数据类型

2.4 运算符、表达式与计算

2.5 数据输出与简单数据输出程序

2.6 数学函数及其使用

2.7 Dev-C++中的辅助编辑功能

2.6 数学函数库及其使用

2.6.1 函数、函数调用

- 标准库提供了许多函数，实现许多功能。
- 其中有一组数学函数，实现常用数学函数计算。
- 要用函数，需要知道名字、使用方式、能完成的工作。不必关心功能如何实现。
- 提供这些常用功能是为了写程序方便。
- 使用标准库数学函数，程序前面必须写：

#include <cmath>

（告诉编译程序去读取文件 **cmath**，从而知道有那些数学函数可以使用。否则编译程序不认识这些数学函数，就会报错“**xxxx** 函数在此范围内没有声明”）



- 标准库中的数学函数对实参的个数和类型，以及返回值类型都有明确说明。
- 对函数的类型特征通常以如下格式进行说明（以 **sin** 函数为例）：



说明格式：

返回值类型

函数名

参数表

double sin(double)

示例解释：

计算后返回一个
double 类型的数值

需要一个 **double**
类型的实参
数学含义为弧度值

- 标准库函数名都由小写字母拼写。

- 函数的使用形式：

函数名(实参, ..., 实参)

- 函数通常规定了需要几个实参和实参类型。
- 要根据规定来写出函数调用表达式。



double sin(double)

- 用户按照上述类型特征说明写出调用表达式：

提供一个double类型的实参

1 + sin(2.4)

返回值为double类型，参与后续计算

【例2-6】求两邻边长为 3.5 和 4.72 米，两边夹角为 37 度的三角形的面积。

$$S = \frac{1}{2} ab \sin \theta$$

```
#include <iostream>
#include <cmath>
using namespace std;
```

```
int main () {
    cout << "Area of the triangle: ";
    cout << 3.5 * 4.72 * sin(37.0 / 180 * 3.1415926) / 2
        << " m^2\n";
    return 0;
}
```

不能写成：

1/2*3.5*4.27 *sin(弧度值)

角度转换为弧度

不能写成：**37/180***3.1415926

- 主要数学函数（大部分的特征与**sin**一样）：

sin cos tan asin acos atan sinh cosh

tanh exp log log10 sqrt fabs

乘幂：**double pow(double, double)**

实数余数：**double fmod(double, double)**

阅读并理解它们的类型特征，写出调用语句。

例如：**3 + pow(2.5, 3.4)**

学习把数学公式写成 **C /C++** 语言表达式。例：

cos 1.5

cos(1.5) //注意要写括号

cot 2.3

1/tan(2.3) //用 **tan** 表示 **cot**

ln 3.6

log(3.6) //注意要写括号

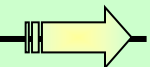
log 4.7

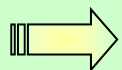
log10(4.7)

log₅ 5.8

log (5.8)/log(5)

或 **log10 (5.8)/log10(5)**





- 续上表（把数学公式写成C语言表达式）

$$\sqrt{\pi + 1}$$

sqrt(3.1416+1)

$$e^{\sqrt{\sin \pi + 1}}$$

exp(sqrt(sin(3.1416)+1))

$$\arctan(\log_3 e)$$

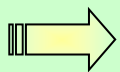
atan(log(2.7183)/log(3))

$$|\pi^2 - 8.7|$$

fabs(3.1416*3.1416 - 8.7)
//平方通常写成连乘，而不使用 **pow** 函数

$$10 + 2.5^{3.4}$$

10 + pow(2.5, 3.4)



把数学公式写成C/C++语言表达式 **注意事项:**

- 多级括号要正确配对。（否则编译时会报错）
- 数学函数名与相应的 C/C++ 语言函数名有差别：
 **$\arcsin \leftrightarrow \text{asin}$, $\arctan \leftrightarrow \text{atan}$, $\ln \leftrightarrow \text{log}$,
 $\log \leftrightarrow \text{log10}$,**
- 手工简写函数要转换为相应的函数：
 $5^3 \rightarrow 5*5*5$, $5^{6.4} \rightarrow \text{pow}(5, 6.4)$

2.5.3 函数调用中的类型转换

例：sin函数的类型特征为 **double sin(double)**

- 函数对参数有类型要求，实参表达式有类型。
- 规定：实参类型与函数要求不符时，表达式值自动转为函数要求类型的值后再送给函数。

例：下面表达式计算中将出现两次类型转换：

sin(2) * sin(4)

问：设 **func** 的类型特征为 **int func (int)**。如下表达式计算会出现类型转换（几次/哪里/什么？）：

4 * func(3 * 2.7)

【例2-7】简单级数计算：计算 $\sum_{n=1}^{10} \sin \frac{1}{n}$ 的值。

● 有问题的程序：

```
#include <iostream>
#include <cmath>
using namespace std;
```

```
int main() {
    cout << "result: " << sin(1) + sin(1/2) + sin(1/3)
        + sin(1/4) + sin(1/5) + sin(1/6) + sin(1/7)
        + sin(1/8) + sin(1/9) + sin(1/10);
    return 0;
}
```

如何改正？

答：全部改成 **sin(1.0/2)** 的形式。

例2-6: 三角形三边长为3、5、7厘米，求其面积。

```
#include <stdio.h>
#include <math.h>
int main () {
    printf("%f\n",
        sqrt(((3+5+7)/2.0) *
            ((3+5+7)/2.0 - 3) *
            ((3+5+7)/2.0 - 5) *
            ((3+5+7)/2.0 - 7) )
    );
    return 0;
}
```

$$s = (a + b + c) / 2$$

$$S = \sqrt{s(s-a)(s-b)(s-c)}$$

注意分析计算表达式时的类型转换

在处理具体问题时，要同时考虑以下几点：

1. 数学需求； 2. 类型转换； 3. 输出格式。

例如，本章练习题第5大题第**(9)**题：

$$1 + \frac{2}{3 + 4/5}$$

有同学写出如下语句：

```
printf("ex2-5-9: %f \n", 1+2/(3+4/5));
```

错误原因：

(1) 数学上实际是需要计算含小数的结果！

(2) 表达式 $1+2/(3+4/5)$ 的结果是整型值 **1**，试图以 **%f** 格式输出时会严重出错。



要同时考虑多个因素才能正确解决。

$$1 + \frac{2}{3 + 4/5}$$

以下语句是否正确解决问题？

`printf("ex2-5-9: %f \n", 1. + 2 / (3 + 4 / 5));` ×

`printf("ex2-5-9: %f \n", 1 + 2. / (3 + 4 / 5));` ×

`printf("ex2-5-9: %f \n", 1 + 2 / (3. + 4 / 5));` ×

`printf("ex2-5-9: %f \n", 1 + 2 / (3 + 4. / 5));` ✓

`printf("ex2-5-9: %f \n", 1 + 2 / (3 + 4. / 5.));` ✓

`printf("ex2-5-9: %f \n", 1. + 2. / (3. + 4. / 5.));` ✓

`printf("ex2-5-9: %d \n", 1. + 2. / (3. + 4. / 5.));` ×

`printf(" %g \n", 1 + 2 / (3 + (double)4 / 5));` ?

2.6.3 inf 与 nan

- 有些数学计算中对参数的值有要求，例如不能用 **0** 作为除数，在实数范围内不能对负数求开方或求对数。
- 在计算机的计算过程中同样也需要满足这些要求。如果在程序中写出了这样不符合数学计算要求的表达式，程序运行时对这些表达式求值所得结果会表示为 **inf** 或 **nan**。
- “**inf**”(infinite)表示“**无穷大**”。一般是因为**表达式求值结果超出浮点数的表示范围**。常见的原因是在表达式中出现了以 **0** 作为除数，或者是出现了浮点数溢出。
- “**nan**”(Not a number)表示“**无效数字**”。一般是因为**对浮点数进行了未定义的操作**，如对负数求开方或求对数（**C**和**C++**语言中的数学计算默认在实数范围内进行）。
- 如果出现了“**inf**”或“**nan**”，则意味着程序中有不符合数学计算要求的表达式，编程者应该检查程序并排错。

第 2 章 数据与简单计算程序

2.1 计算机中的数值表示与存储

2.2 基本字符、名字表示、标识符和关键字

2.3 常用数据类型

2.4 运算符、表达式与计算

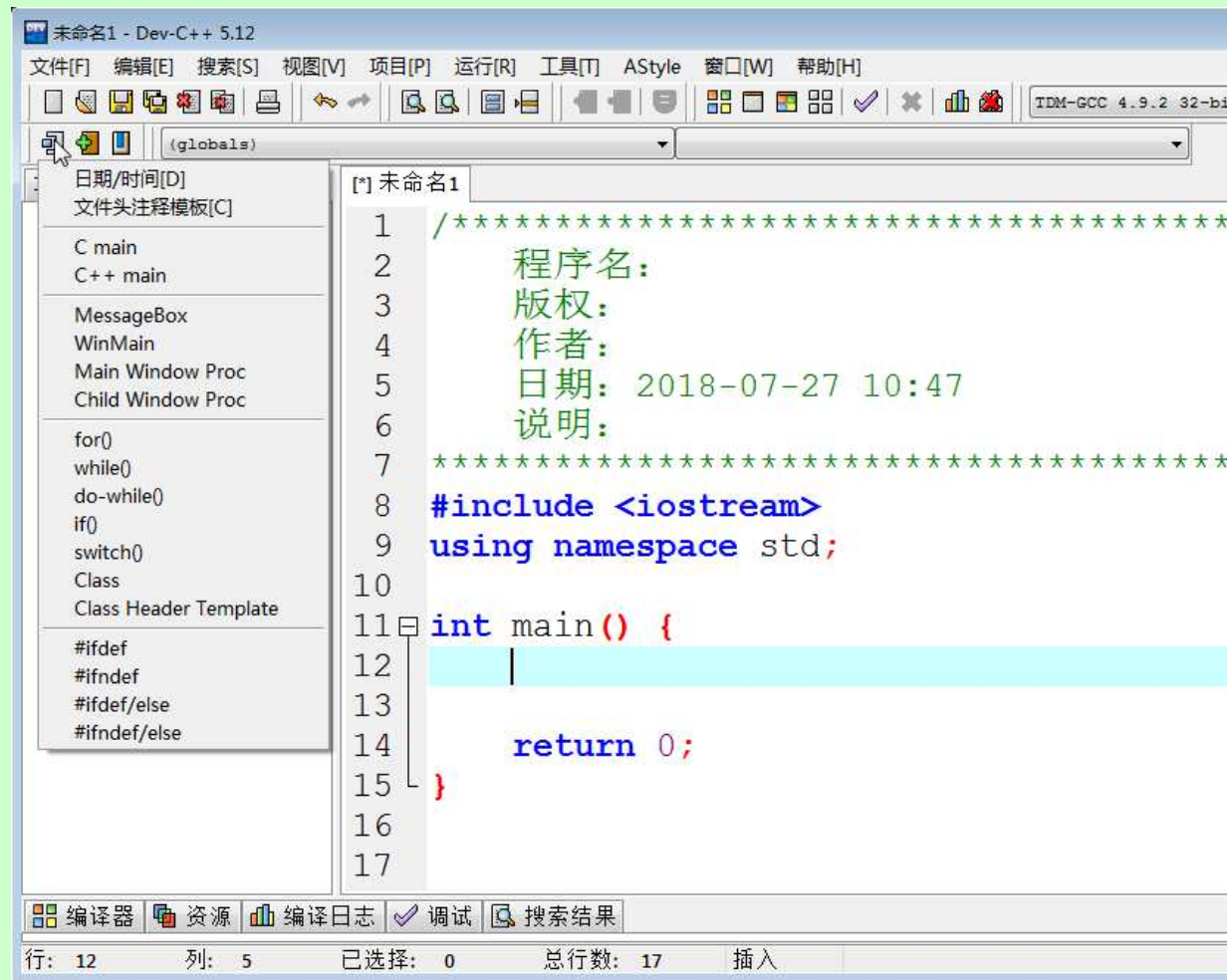
2.5 数据输出与简单数据输出程序

2.6 数学函数及其使用

2.7 Dev-C++中的辅助编辑功能

2.7 Dev-C++中的辅助编辑功能

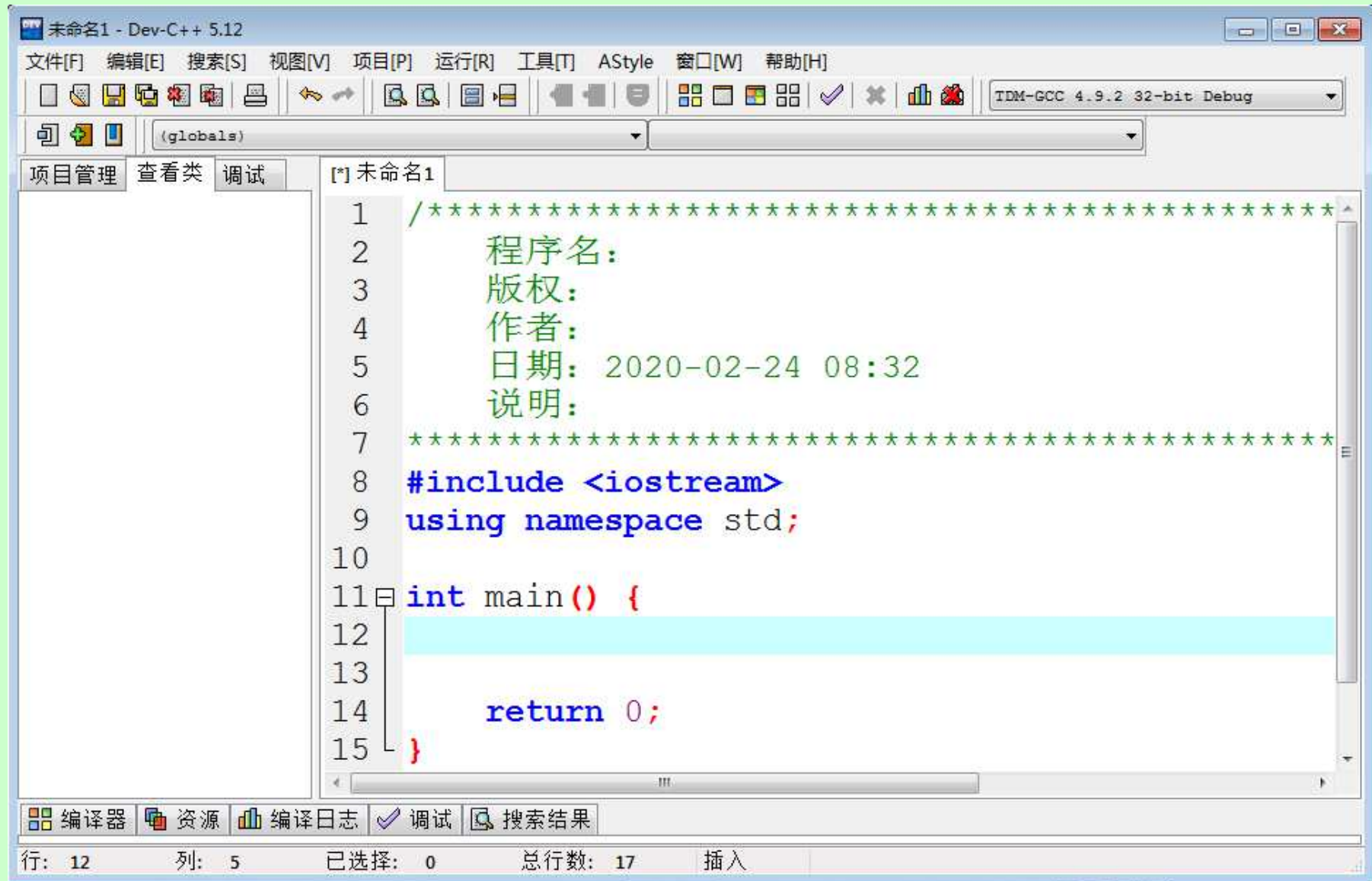
- 插入常用的文字或代码模板



常用菜单功能：

- “文件”：“另存为” “最近用过的文件”
- “编辑”：“注释”（快捷键是**Ctrl+.**）和“取消注释”（快捷键是**Ctrl+,**）
- “搜索”：“搜索...” “替换...”

- 熟记Dev-C++界面各个功能按钮:



工欲善其工，必先利其器

提高操作速度，熟练掌握**Dev-C++**的快捷键：

- 切换输入法：
- 新建源程序：
- 保存文件：
- 查找：替换：
- 设为注释：取消注释：
- 编译：运行：
- 编译运行：